Meta-Reasoning Algorithm for Improving Analysis of Student Interactions with Learning Objects using Supervised Learning

L. Dee Miller and Leen-Kiat Soh
Department of Computer Science and Engineering
University of Nebraska, Lincoln
{lmille, lksoh}@cse.unl.edu

ABSTRACT

Supervised learning (SL) systems have been used to automatically learn models for analysis of learning object (LO) data. However, SL systems have trouble accommodating data from multiple distributions and "troublesome" data that contains irrelevant features or noise—all of which are relatively common in highly diverse LO data. The solution is to break up the available data into separate areas and then take steps to improve models on areas containing troublesome data. Unfortunately, finding these areas in the first place is a far from trivial task that balances finding a single distribution with having sufficient data to support meaningful analysis. Therefore, we propose a BoU meta-reasoning (MR) algorithm that first uses semi-supervised clustering to find compact clusters with multiple labels that each support meaningful analyses. After clustering, our BoU MR algorithm learns a separate model on each such cluster. Finally, our BoU MR algorithm uses feature selection (FS) and noise correction (NC) algorithms to improve models on clusters containing troublesome data. Our experiments, using three datasets containing over 5000 sessions of student interactions with LOs, show that multiple models from BoU MR achieve more accurate analyses than a single model. Further, FS and NC algorithms are more effective at improving multiple models than a single model.

Keywords

Learning Object Analysis; Supervised Learning; Clustering; Meta-Reasoning

1. INTRODUCTION

Learning objects (LOs) are independent and self-standing units of learning content that are predisposed to reuse in multiple instructional contexts [2]. An example of an LO is a self-contained lesson on recursion with a tutorial, interactive exercises, and assessment questions. In general, the analysis of student interactions with LOs is important for many groups including students, instructors, researchers, and content developers [16]. First, for students, such analyses can improve student study strategies and allow for more self-regulated learning [1]. Second, instructors can use such analyses to choose appropriate LOs for their students [8]. Third, such analyses can help researchers and content developers investigate which student interactions are associated with the different learning outcomes [6].

One previously used approach for the analysis of student interactions with LOs is supervised learning (SL) systems [16]. SL systems learn a model from previously recorded sessions of student interactions (features) and learning outcomes (labels) that can predict the learning outcome for a specific session of student interactions with a high degree of accuracy.

SL systems have one main advantage over other approaches (e.g., statistical analysis): they learn the model automatically without the need for direct human intervention. First, learning the model can help students and instructors. Such a model predicts the learning outcome for a student in real-time based on the observed student interactions [15]. Such predictions can allow the LO to adjust the content presented to a student while he or she is taking the LO and provide real-time updates to instructors on student mastery of LO content. Second, a model learned automatically without human intervention provides independent, high-level guidelines on which types of student interactions are associated with the learning outcomes [4]. Such guidelines can serve as a useful starting point for further investigation by researchers and inform content developers on which parts of the LO may need to be revised.

However, SL systems have some potential problems which can limit the effectiveness of their models for analysis:

First, SL systems assume that the training data from previously recorded sessions comes from a single underlying distribution. Unfortunately, such training data is likely to come from multiple distributions and be highly diverse due to a wide variety of factors including students with different backgrounds, LOs with different content, instructors providing varying amounts of support for the LO content, etc. These factors make it ***difficult to learn a single model which can "fit" all this highly diverse training data*** and still achieve high accuracy.

Second, SL systems assume that the training data available is relatively "clean" being free of student interactions unrelated to the learning outcomes (i.e., irrelevant features) and errors in the student interactions and learning outcomes provided (i.e., noise). Unfortunately, such training data is all too likely to contain both irrelevant features and noise. Irrelevant features are relatively common when researchers are uncertain which student interactions are relevant and, thus, record as many student interactions as possible since they cannot retroactively record additional interactions. Noise is relatively common when developers fail to create assessment questions appropriate for all students and when students are motivated to "game the system" to achieve a certain learning outcome (e.g., a good grade on the LO). These factors make it ***difficult for a single model to achieve high accuracy on areas containing large amounts of "troublesome" training data*** with irrelevant features and/or noise.

Intuitively, we could address these problems and improve the effectiveness of SL systems for analysis by using an approach which first breaks up the training data into areas—each containing similar student interactions—and learns a separate model on each area. We could then identify which areas consistently contain "troublesome" training data and take steps to improve the models in those areas.

However, breaking up the training data and finding areas with "troublesome" training data are far from trivial tasks. As previously mentioned, the training data collected is likely to be highly diverse. Such diversity makes it difficult for an approach to find suitable areas balancing two factors. First, each area should contain similar data from a ***single distribution***. Second, each area should contain sufficient training data to support ***meaningful analysis***. Further, the model learned on an area is likely to fit the irrelevant features and/or noise in that area which can make it difficult to identify the areas containing ***troublesome training data***.

Therefore, we propose new meta-reasoning algorithm called the Boundary of Use (BoU MR) to improve the effectiveness of SL models for analysis of student interactions with LOs. Our algorithm first uses an iterative process, based on semi-supervised clustering, which breaks up the training data in different ways until suitable areas are found. These suitable areas, which we dub BoU clusters, include training data with similar student *interactions* from a ***single distribution*** which, nevertheless, have multiple learning outcomes to support ***meaningful analysis***. After clustering, the BoU MR learns a separate model for each of these clusters. Then, our algorithm evaluates each of these BoU clusters using a localized estimate to detect ***troublesome training data*** (e.g., feature selection for irrelevant features). Finally, our algorithm takes steps to selectively improve the models for the difficult BoU clusters containing troublesome training data; for example, removing irrelevant features and relearning the model on the "refined" training data.

In the following, we will investigate the BoU MR using two objectives. Objective 1 is to investigate the impact of breaking up the training data on LO datasets using three types of SL systems to learn the models: decision trees, support vector machines, and artificial neural networks. We compare the accuracy for a single model with that for multiple models from the BoU MR. Objective 2 is to investigate the effectiveness of BoU MR for improving its models on difficult BoU clusters. For this objective, we consider both feature selection and noise correction algorithms. We compare the accuracy to the BoU MR that uses these algorithms to help selectively relearn the models for difficult BoU clusters to a single model relearned after the same algorithm is applied to all the training data.

The rest of the paper is organized as follows. Section 2 provides background on SL systems and model improvement algorithms used in our study. Section 3 describes our BoU meta-reasoning algorithm in more detail. Section 4 discusses the experimental setup and results. Finally, we conclude and outline future work.

2. BACKGROUND

Here we discuss background on the SL systems and model improvement algorithms. Discussion of the LO datasets is deferred until Section 4.

2.1 Supervised Learning (SL) Systems

We consider three types of SL systems in the experiments below. To help demonstrate the effectiveness of our meta-reasoning algorithm for SL systems in general, we chose three widely used SL systems with very different properties. First, *artificial neural networks* (ANNs) learn a vector of weights on features in the dataset to choose the labels for new data [19]. ANNs consist of multiple nodes connected to threshold functions or to additional layers of nodes. ANNs are updated iteratively (e.g., using gradient descent) until they correctly predict the labels for the training data. Second, *decision trees* (for classification) learn a tree data structure to generate the labels for new data [19]. The decision tree first selects one feature as the root node and adds an edge for every label value. The decision tree continues to add nodes and edges recursively until all the training data has been sorted into groups with similar labels. The leaves are then set to the common label. Third, *support vector machines* (SVMs) learn a hyperplane to separate the training data such that data on the same side mostly have the same label [19]. SVMs first use a kernel function to transform all values for the dataset into higher dimensional space where they are linearly separable. Then, the SVM attempts to maximize the distance (i.e., margin) between the training data with different labels.

2.2 Model Improvement Algorithms

We consider two types of algorithms for improving the models in the experiments below: feature selection and noise correction.

First, feature selection (FS) algorithms find the subset of relevant features for the dataset using an evaluation criterion based on *filters* or *wrappers*. Filters evaluate the relevant features using only the intrinsic properties of the data whereas wrappers use the accuracy of the SL system model [10]. To avoid overfitting common to wrapper-based feature selection, we use a state-of-the-art filter-based FS algorithm called Lasso in the experiments below. Lasso FS uses a shrinkage method for FS which maintains a coefficient for each of feature (Hastie et al., 2011). Lasso computes these coefficients by using a pairwise coordinate descent approach to minimize the sum of squares subject to a constraint on the coefficients. Features whose coefficients have shrunk to zero are considered to be irrelevant and removed entirely from the dataset.

Second, noise correction (NC) algorithms are designed to identify noisy labels and then remove or replace them. There are two general types of noise correction algorithms [13]: (1) noise *tolerant* correction modifies existing SL systems to better accommodate noisy labels (e.g., rule-post pruning for decision trees) and (2) noise *filtering* detects noisy labels in the training data before the model is learned. We use both types of noise correction algorithms in the experiments below. We use decision trees rule post-processing and SVMs with soft margins both designed to accommodate noisy labels [19] for the former and a state-of-the-art algorithm called LSVM [18] for the latter. LSVM uses a hybrid approach for NC that starts by using a *k*-Nearest Neighbor algorithm to select the neighborhood of similar data around a given instance. LSVM then learns a local SVM on that local neighborhood (hence the name). Based on the maximal margin principle, if the LSVM incorrectly predicts the label for that instance, then the label is deemed noisy. Furthermore, to avoid accidently injecting noise into the training data, in our experiments, noisy labels are removed rather than being replaced.

3. METHODOLOGY

The central idea for the BoU meta-reasoning (MR) algorithm is to first break up the training data into special BoU clusters containing sessions with similar student interactions (i.e., instances) from a single distribution. At the same time, the cluster should sufficient data with multiple learning outcomes (i.e., labels) to support a meaningful analysis in the form of a SL model. In a very real sense, BoU clusters allow us to "zoom in" and get a more detailed analysis on highly diverse LO data than could be obtained using all the data together.

Next, our algorithm learns a separate model based on the training data in each BoU cluster. By using only the data in a single cluster, BoU MR guarantees that each model is more detailed and expressive on member data than a single model trained on all data together. After learning the separate models, our algorithm uses a localized estimate of the accuracy for each model by comparing the predicted and actual labels for the cluster members: ***correct*** *when the predicted matches the actual; otherwise,* ***incorrect***. Based on the predominant correct/incorrect label, BoU MR thus assigns each BoU cluster a type: correct BoU clusters where the model is doing well and incorrect clusters where the model is struggling on troublesome data.

Figure 1 provides an example of four such BoU clusters. In this figure, the clear circles are correct training data instances and grey circles are incorrect instances. Clusters 1 and 2 contain only correct instances and are thus flagged as correct. Note that Cluster 1 contains only a single label while Cluster 2 contains both labels—both are considered correct based on the localized estimate. Clusters 3 and 4 are flagged as incorrect since they contain a mix of both correct and incorrect instances.

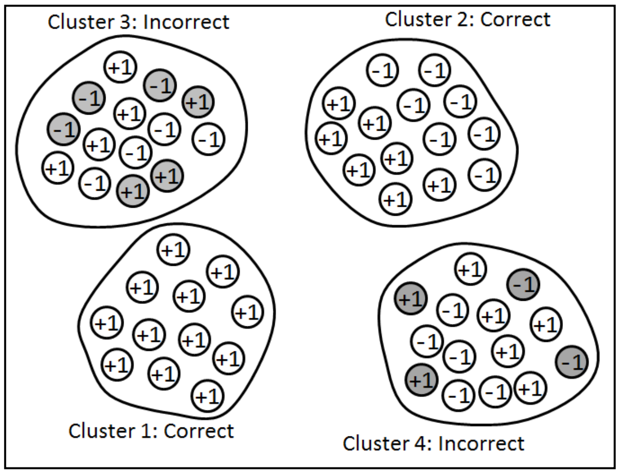

Figure 1. Example BoU Clusters. The grey data instances are those on which the model failed to predict the correct label.

After identifying the aforementioned clusters, BoU MR takes steps to improve the models for incorrect clusters. This is done by using either a feature selection or noise correction algorithm *selectively on only the incorrect clusters*. In this way, the models are left alone on correct clusters where they are already doing well. The BoU MR then relearns each model for a previously incorrect cluster using the refined data in that cluster.

Taken together, the combination of multiple, expressive models learned on highly diverse LO data and selective improvement on clusters containing troublesome data allows our BoU to improve the overall effectiveness of SL models for analysis of student interactions with LOs.

3.1 BoU Essential Components

Here we discuss the basic process and equations used for creating the clusters and deciding whether they are correct or incorrect. First, to make use of the BoU notion of clusters to find clusters with a single distribution that support meaningful analysis, we use a semi-supervised clustering (SSC) algorithm [9] to cluster the training data. Briefly, SSC algorithms create clusters based on both similarity in the training data and additional information available on how the session instances should be clustered (e.g., constraint that two instance must/cannot be clustered together). For our purpose—to find BoU clusters, the additional information that we incorporate for each session instance is whether or not the model predicts the label correctly or incorrectly. The actual SSC algorithm used is based on the *k*-Means variant discussed in Kulis et al. [9]. The modified objective function for BoU-style clusters π can be expressed as:

$$\sum_{c=1}^{k} \sum_{x_i \in \pi_c} \|x_i - m_c\|^2 + \sum_{\substack{x_i, x_j \in C \\ s.t. l_i \neq l_j}} w_{ij} \tag{1}$$

where C is the set of cannot-link constraints, w_{ij} is the penalty cost for violating a constraint involving points x_i and x_j and l_i refers to the model prediction for x_j s.t. $l_i \in \{\text{correct}, \text{incorrect}\}$. The first term in Eq. 1 is the *k*-Means objective function that chooses the closest centroid while the second term is a penalty function for assigning a data instance deemed correct to a cluster with incorrect instances (or vice-versa). The training data is assigned to the cluster that *minimizes* this objective function. This predisposes the SSC algorithm to find high similarity clusters that have either predominately correct or incorrect instances. Note that clusters with predominately incorrect labels are guaranteed to contain *multiple labels* since a single-labeled cluster would be trivial for the model.

Second, the BoU needs to be able to evaluate each cluster to decide whether the model for that cluster needs improvements. Here we propose using a localized estimate of model performance to decide whether the model needs improvement to accommodate troublesome data. This estimate can be expressed as:

$$est(\pi_c) = \sum_{x_i \in \pi_c} [l_i = \text{correct}] \,/\, |\pi_c| \tag{2}$$

where π_c is the cluster under consideration, x_i is the cluster member, and l_i refers to the model prediction.

Third, we decide whether each BoU cluster is *correct* or *incorrect*. The decision making strategy here is to make use of a specified confidence interval to identify correct clusters where improvement is not needed:

$$type(\pi_c) = \begin{cases} \text{correct } if\, est(\pi_c) \geq 1 - \delta \\ \text{incorrect } otherwise \end{cases} \tag{3}$$

where est is the localized estimate (Eq. 2) and δ is the purity threshold parameter for the confidence interval. Eq. 3 is based on work in Dasgupta & Hsu [5] where clusters are evaluated using confidence intervals on the correctly-labeled member data to decide whether to request further labels for the member data.

Finally, we also create a hierarchy of BoU clusters. The BoU uses a hierarchical, top-down approach that iteratively (1) splits the data into clusters and identifies the correct and incorrect clusters, and (2) selectively improves the models on incorrect clusters. Specifically, at each layer of the cluster dendrogram, the SSC algorithm previously described splits the data into BoU clusters (Eq. 1). Next, each cluster from the split is assigned a type (Eq. 3) based on the localized estimate (Eq. 2). If the cluster is deemed incorrect, the specified improvement algorithm (cf., Section 2.2) runs using only the member data in that cluster and the model is relearned on that refined data. Correct clusters skip both steps and instead inherit the model from their parent cluster. After BoU algorithm stops splitting, the clusters and relearned models at the leaves of the dendrogram are used to make up the ensemble that predicts the labels for new data. The leaves are used so that each training instance belongs to only a single cluster thus avoiding confusion on cluster membership. Ultimately, this ensemble predicts the labels for a new instance by selecting the cluster containing the most similar instances and then using the model associated with that cluster to predict the label.

An example of this top-down approach is given in Figure 2 with the type, improvement, and model for each cluster. In this example, we use FS as the model improvement algorithm that gives a set of relevant features (f). This figure shows how the clusters are split and the models are inherited or relearned from one layer to the next. The original cluster starts with the model learned on all the data. After the improvement ($f1$) and relearn steps ($m1$), this cluster is split into correct and incorrect clusters. *The correct cluster uses the model from the parent*, while the incorrect cluster goes through model improvement and relearning before being split again. As shown in the last split, one child can retain the parent model ($m3$) while the other goes through the improvement and relearn steps.

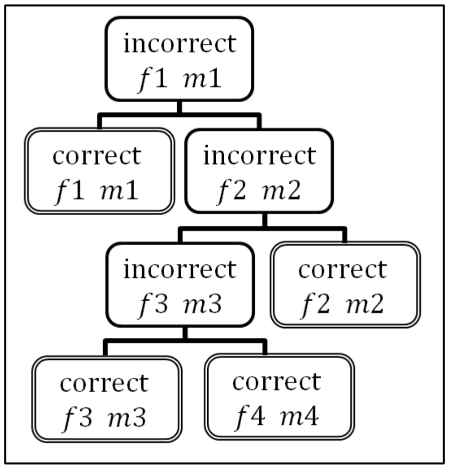

Figure 2. Cluster Splits from our BoU MR along with Type, FS Improvement (f), and Model (m). The double width borders denote the final set of clusters used.

3.2 BoU MR Algorithm

Here we present the complete BoU MR algorithm, as shown in Figure 3. Before we begin, there are two general guidelines we adopt to decide when to stop splitting the clusters. First, to avoid breaking up data in a single distribution, we stop splitting when a correct cluster is found with a high purity in terms of correctly-labeled instances (i.e., *purity* stop). Second, to support meaningful analysis, we stop splitting when clusters lack sufficient coverage (based on the percentage of the training data they contain) to learn a detailed model (i.e., *coverage* stop).

Our algorithm starts with a single cluster with all the training data and a model trained on that data. The algorithm runs recursively to create the dendrogram of BoU clusters. First, this algorithm uses Eq. 3 to compute the type for the BoU cluster (line 1). If the type is incorrect, the specified improvement algorithm is used on that BoU cluster's member data (cf., Section 2.2) and its model is relearned (lines 2-3). Subsequently, the cluster's type is updated based on the relearned model (line 4). If a cluster's type is now correct, then there is a purity stop and the algorithm returns the BoU cluster and its relearned model. Otherwise, the algorithm splits the training data into two new BoU clusters using the SSC algorithm previously discussed (Eq. 1). If both the new clusters meet the minimum coverage requirement (line 6), containing a percentage of the training data above the threshold φ, then the algorithm runs recursively on the two new BoU clusters with the parent's model. Otherwise, there is a coverage stop due to insufficient instances and the parent cluster/model is returned.

The BoU MR algorithm runs in polynomial time based on the number of recorded sessions and, as such, runs fast even as the number of LOs increases. The actual time complexity is dependent on the clustering algorithm: *O(IDF)* where *I* is the max iterations, *D* is the number of session instances, and *F* is the number of features. Further, the actual BoU clusters and the SL models can be computed offline to accommodate thousands of LOs. The real-time analysis only consists of mapping the new session to the BoU cluster based on current student interactions with the LO. This can be done very quickly as the number of clusters is much less than the total number of recorded sessions.

Some readers may argue that, by looking at the way the clusters are identified hierarchically, we are actually introducing overfitting on the data when creating the BoU clusters. But recall that the BoU MR algorithm is designed to prevent the labels from having too much influence on the BoU clusters: its clusters are created based on both feature similarity and labels, and not just labels alone. Additionally, the coverage stop also helps in this regard by acting as a regularizer and rejecting small clusters.

```
$C = Single\ cluster\ with\ all\ the\ training\ data$
$m = Model\ trained\ on\ C$
$I = Model\ improvement\ algorithm$
$S = Supervised\ learning\ system$
$\textbf{function}\ BoUClustering(C, m)\ \textbf{returns}\ C'\ and\ m'$
(1)  $\textbf{if}\ type(C) == incorrect$ // check purity stop
(2)      $C' \leftarrow improve(C, I)$
(3)      $m' \leftarrow relearn(C', S)$
(4)      $\textbf{if}\ type(C') == incorrect$ // check purity stop
(5)          $C_1, C_2 \leftarrow SSC(C', 2)$ // split the cluster
(6)          $\textbf{if}\ coverage(C_1) > \varphi\ \textbf{and}$
                 $coverage(C_2) > \varphi$ // check coverage stop
(7)              $BoUClustering(C_1, m')$
(8)              $BoUClustering(C_2, m')$
(9)          $\textbf{end if}$
(10)     $\textbf{end if}$
(11) $\textbf{else}$
(12)     $C' \leftarrow C$
(13)     $m' \leftarrow m$
```

Figure 3. BoU MR Algorithm.

4. IMPLEMENTATION AND RESULTS

Here we start by describing the experimental setup including the learning object (LO) datasets. In Section 4.1, we provide results demonstrating the effectiveness of using the BoU to learn multiple models on the LO datasets (Objective 1). In Section 4.2, we provide results for using the BoU to improve existing models with feature selection and noise correction algorithms (Objective 2).

First, we use three widely studied SL systems in the experiments below: artificial neural networks (ANNs), support vector machines (SVMs), and decision trees (DTs). We use the Java implementations for all three from the Weka library with the parameters values suggested in Witten et al. [19]. The BoU MR algorithm uses a Java implementation for the semi-supervised clustering algorithm based on Kulis et al. [9]. We use of 0.1 for the purity threshold (δ) and a 0.1 for the coverage threshold (φ) both fine-tuned based on empirical results. Additionally, this clustering algorithm normalizes the student interactions features to the same range before creating the clusters. For the model improvement algorithms, we use a Java implementation of Lasso feature selection based on the R glmnet package [7]. We use the C++ implementation of LSVM noise correction from the FaLKM-lib package [17]. We use the values suggested in Segata [17] for the numerous parameters for LSVM.

Second, the intelligent learning object guide (iLOG) LO datasets used in the experiments are based on a three year deployment of 16 learning objects to introductory CS courses at the University of Nebraska, Lincoln [11]. During this deployment, there were over 5000 separate sessions between students and LOs. Large amounts of data were collected including (1) student interactions with the LOs during the tutorial, exercise, and assessment components (e.g., time spent on a page), (2) student demographic data (e.g., gender), (3) scores on the CS placement test [12], and (4) survey responses to both MSLQ and evaluation Likert surveys. The iLOG datasets distill the data collected into the instances, features, and labels necessary for supervised learning. Each instance represents a student-LO session with the features summarized in Table 1. The label for an instance is whether or not students passed the LO assessment component (i.e., if a student achieves $\geq$ 70% then she passes, otherwise she fails). As shown in Table 1, there are relatively few changes in the features collected from one year to the next. The increase in instances is the result of deployment to a larger number of courses.

We use the iLOG datasets in the experiments because they exemplify the aforementioned problems with SL systems in the following manner. First, the LOs were deployed to students in introductory CS courses with highly diverse backgrounds (e.g., CS majors, nonmajors, etc.) resulting in *multiple distributions* in the resulting datasets. Second, the LOs were deployed online using the Moodle Learning Management System and students were required to take the LOs and part of their course grades (5%). The difficulty of writing LO content suitable for all students and the online deployment, taken together, resulted in both *irrelevant features and noise* in the data tracked. For example, students trying to achieve high assessment scores without spending time on the tutorial content. Thus, these datasets are prime candidates for using FS and NC.

Table 1. Summary of the iLOG datasets in the experiments.

Features	**iLOG 2008**	**iLOG 2009**	**iLOG 2010**
Metadata	5	5	5
Tutorial	10	10	10
Exercises	20	20	16
Assessment	10	10	10
Student Demo	9	9	9
Placement	16	16	16
MSLQ	47	45	50
Evaluation	10	9	9
Total	127	124	125
Instances	**iLOG 2008**	**iLOG 2009**	**iLOG 2010**
Fail	426	738	1228
Pass	604	1131	2215
Total	1030	1869	3443

Finally, the experiments below compare the single model with the multiple models from the BoU. For all experiments, we provide both the test and F1 accuracy results based on ten-fold cross validation. In Section 4.1, we compare a single model to the BoU models learned using three SL systems (ANN, SVM, and DTs). In Section 4.2, we compare a single model to the BoU models after refining the data using FS and NC. This results in six (FS or NC × ANN, SVM, or DT) configurations.

4.1 Multiple Model Investigation

Table 2 provides the test and F1 accuracy, on the iLOG datasets, for single model and the BoU multiple models. The BoU multiple models provide higher test and F1 accuracy than a single model on all three iLOG datasets. These results are reasonable given that the BoU can break up the iLOG dataset to better accommodate data from multiple distributions, for example, LOs deployed to different courses, students with different majors, etc.

To probe further into how the BoU multiple models accommodate the iLOG data, Figure 4 provides the actual decision trees on the iLOG 2008 dataset created using a single model and multiple models based on three clusters. (The trees for the iLOG 2009 and 2010 datasets are similar.) As shown in Figure 4, the trees learned on Clusters 1-2 have a very different idea of what features are the most important for predicting the labels (i.e., at the root) than the a single tree learned on all the data. By using these diverse trees—i.e., models, the BoU MR can better model the separate distributions in the iLOG datasets than forcing a single model to accommodate all the data. At the same time, the BoU clusters contain sufficient data to allow for fully expressive trees on the iLOG dataset. The proof of this is that Cluster 3 actually learns the same tree as the single model. Taken together, the capability to find diverse trees while retaining the same tree as the single model helps explain the BoU MR benefits to the test and F1 accuracy results. Additionally, BoU models learned on the local data provide more specific and detailed analyses than using a single model on all the data. These analyses can uncover very interesting connections in the data that would otherwise be hopelessly buried in the single model such as that between evaluation survey questions and gender in Cluster 1. Overall, these results help establish the effectiveness of BoU multiple models for LO analysis.

Table 2. Test and F1 accuracy for a single model and BoU multiple models. Grey cells indicate higher test accuracy while (*) indicates significantly higher accuracy (*t*-test, *p* <= 0.05). The average number of BoU clusters is also given.

Dataset	Test Accuracy		F1 Accuracy		Ave. #
iLOG 2008	Single	BoU	Single	BoU	Clusters
ANN	0.69	0.74*	0.63	0.67	1.90
SVM	0.68	0.73*	0.64	0.69*	2.10
DT	0.72	0.73	0.68	0.68	4.60
iLOG 2009	Single	BoU	Single	BoU	Clusters
ANN	0.69	0.74*	0.60	0.67*	1.90
SVM	0.67	0.69*	0.58	0.61*	2.00
DT	0.62	0.65	0.52	0.56	2.00
iLOG 2010	Single	BoU	Single	BoU	Clusters
ANN	0.70	0.72*	0.56	0.61	3.20
SVM	0.69	0.71	0.57	0.62*	3.60
DT	0.65	0.67*	0.50	0.51	2.30

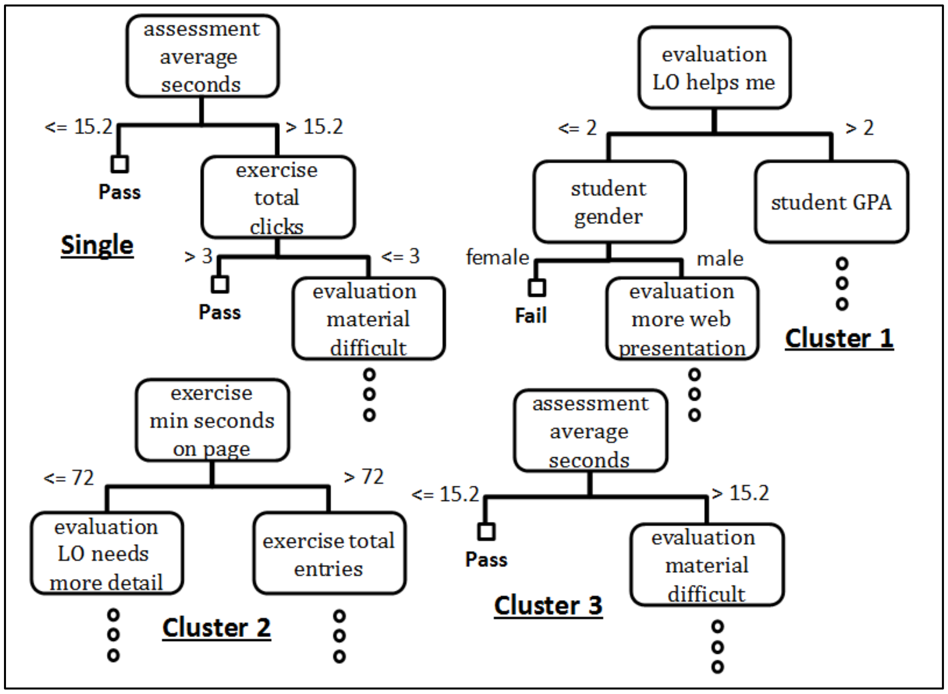
assessment average seconds
<= 15.2
> 15.2
Pass
exercise total clicks
Single
> 3
<= 3
Pass
evaluation material difficult
evaluation LO helps me
<= 2
> 2
student gender
student GPA
female
male
Fail
evaluation more web presentation
Cluster 1
exercise min seconds on page
<= 72
> 72
evaluation LO needs more detail
exercise total entries
Cluster 2
assessment average seconds
<= 15.2
> 15.2
Pass
evaluation material difficult
Cluster 3

Figure 4. Decision trees on the iLOG 2008 dataset created using a single model and multiple models based on BoU clusters.

On the other hand, as shown in Table 2, the accuracy (even for BoU multiple models) is relatively low on all three iLOG datasets (e.g., test accuracy in the 60s for DTs). As alluded to earlier, these datasets contain both irrelevant features and noise both of which are problematic for SL systems in general. In the next section, we show how the BoU MR can use feature selection and noise correction algorithms to break through this ceiling and improve both test and F1 accuracy.

4.2 Model Improvement Investigation

4.2.1 Lasso Feature Selection

Table 3 provides the test and F1 accuracy for the single model and the BoU multiple models both improved using the Lasso feature selection on the iLOG datasets.

Table 3. Test and F1 accuracy for a single model and BoU multiple models both using Lasso feature Selection (FS). Grey cells indicate higher test accuracy while (*) indicates significantly higher accuracy (*t*-test, *p* <= 0.05). The average number of BoU clusters is also given.

Dataset	Test Accuracy		F1 Accuracy		Ave. #
iLOG 2008	Single	BoU	Single	BoU	Clusters
ANN+FS	0.72	0.75*	0.65	0.69*	2.00
SVM+FS	0.72	0.74*	0.66	0.70*	2.00
DT+FS	0.74	0.75	0.70	0.71	2.43
iLOG 2009	Single	BoU	Single	BoU	Clusters
ANN+FS	0.71	0.76*	0.63	0.68*	2.00
SVM+FS	0.71	0.70	0.63	0.62	2.00
DT+FS	0.66	0.69*	0.55	0.60*	2.00
iLOG 2010	Single	BoU	Single	BoU	Clusters
ANN+FS	0.73	0.74	0.58	0.60	2.80
SVM+FS	0.72	0.72	0.61	0.61	3.30
DT+FS	0.67	0.70*	0.52	0.55	2.30

First, we observe that using Lasso on the training data allows nearly across-the-board increases in test and F1 accuracy for both single and BoU multiple models. Additionally, the increases in accuracy reported between Tables 2 and 3 are generally statistically significant (*t*-test, *p* <= 0.05).

To explain, the Lasso feature selection identifies and removes irrelevant features from the training data. Since these features are unimportant to the actual label, had they been incorporated into the model by the SL system, they would tend to confuse and distort the model lowering predictive accuracy and making the model less useful for analysis of student learning outcomes.

Second, using Lasso, we observe that the BoU multiple models still provide generally higher test and F1 accuracy than does a single model on all three iLOG datasets. As previously discussed the BoU retains the capability to break up the iLOG datasets and learn a separate model on each distribution. The BoU also has the capability to further improve and "fine-tune" these models using Lasso selectively only on the clusters that are deemed to contain troublesome data.

To probe further into how the BoU uses Lasso selectively to improve the models, Table 4 provides an example on the iLOG 2009 dataset of the number of relevant features used for the single model and separately for the models in BoU clusters. (The results for the iLOG 2008 and 2010 datasets are similar). As shown in Table 4, for a single model, Lasso removes almost half the features belonging to the exercise and MSLQ categories while mostly retaining features in the other categories. Lasso on BoU clusters gives more diverse results on the features removed. Lasso on cluster 1 removes additional features from the tutorial and assessment categories compared to that for the single model. Next, Lasso on cluster 2 removes a similar number of features as Lasso for the single model. Lastly, the BoU MR does not use Lasso at all on cluster 3 as this cluster is deemed free of troublesome data. Our algorithm prevents Lasso from removing locally relevant features just because they are irrelevant on the rest of the training data. This, in turn, prevents the model from being distorted by removing relevant features necessary for predicting the learning outcome for this cluster. Taken together, *the* *capability to use Lasso selectively*—thus allowing the relevant features to be customized for each cluster—helps explain the BoU benefits to the test and F1 accuracy results. Additionally, Lasso used separately for models provides additional insights that do not show up when Lasso is used for a single model; for example, suggesting that the students with sessions in cluster 1 are getting less out of the LO tutorial and assessment. Overall, these results help establish that BoU can further improve the multiple models for LO analysis using Lasso.

Note that the capability to use Lasso feature selection to improve models is important for educational data mining (EDM) in general since EDM datasets often contain numerous irrelevant features. However, these datasets are also often highly diverse forcing Lasso to make difficult decisions to retain features as relevant that are irrelevant in many areas or remove features because they are only relevant to a minority of the training data. The advantage of using BoU is that its clusters allow for multiple, expressive models. Recall the discussion for Table 4 where Lasso found very different feature vectors when used on different clusters. Combined with the identical decision trees previously discussed (cf., Figure 4), this supports our claim that BoU clusters contain sufficient data to allow for more effective utilization of Lasso separately on the data in each cluster.

Table 4. Number of relevant features selected by Lasso feature run on all the training data (Single) and run separately on the three BoU clusters (C1-C3). The total number of features is also included for reference (2009).

Features	**2009**	**Single**	**C1**	**C2**	**C3**
LO Data	5	5	5	4	5
Tutorial	10	7	3	8	10
Exercises	20	11	9	11	20
Assessment	10	7	6	9	10
Student Demo	9	9	6	6	9
Placement	16	13	9	10	16
MSLQ	45	22	19	24	45
Evaluation	9	9	6	6	9
Total	124	83	63	78	124

4.2.2 LSVM Noise Correction

Table 5 provides the test and F1 accuracy results, on the iLOG datasets, for the single model and the BoU multiple models both using the LSVM noise correction.

Table 5. Test and F1 accuracy for a single model and BoU multiple models both using LSVM noise correction (NC). Grey cells indicate higher test accuracy while (*) indicates significantly higher accuracy (*t*-test, *p* <= 0.05). The average number of BoU clusters is also given.

Dataset	Test Accuracy		F1 Accuracy		Ave. #
iLOG 2008	Single	BoU	Single	BoU	Clusters
ANN+NC	0.73	0.75*	0.65	0.69	2.00
SVM+NC	0.72	0.74	0.68	0.70*	4.10
DT+NC	0.73	0.75*	0.68	0.71	4.00
iLOG 2009	Single	BoU	Single	BoU	Clusters
ANN+NC	0.72	0.75*	0.56	0.67*	2.00
SVM+NC	0.70	0.70	0.50	0.62*	2.00
DT+NC	0.64	0.69*	0.36	0.59*	2.00
iLOG 2010	Single	BoU	Single	BoU	Clusters
ANN+NC	0.71	0.75*	0.52	0.61*	4.10
SVM+NC	0.72	0.70	0.49	0.60	3.50
DT+NC	0.67	0.70*	0.36	0.54*	2.50

First, as with Lasso, using LSVM noise correction allows nearly across-the-board improvements in accuracy. To explain, LSVM improves the model by flagging potentially noisy instances whose labels (i.e., pass/fail) do not match those in nearby instances with similar features. These noisy instances would otherwise distort the model since they provide contradictory labels and, thus, lower predictive accuracy.

Second, using LSVM, on all three datasets the BoU multiple models provide generally higher test and F1 accuracy than a single model. Again, the BoU can "fine-tune" these models using LSVM selectively only on the clusters with troublesome data. Now, as a whole, the total number of instances flagged as noisy by LSVM was comparable when used on the all training data and selectively on the BoU clusters. However, LSVM for multiple models had two advantages. First, by breaking up the training data, the BoU MR simplified the task of distinguishing between instances that actually have noisy labels and those in close proximity to instances with truly different labels. Second, by using LSVM selectively, *our algorithm was able to use LSVM aggressively on clusters where the model was struggling*, such as clusters containing sessions where students tried to game the system, *without worrying about damaging clusters where the model was already doing well by removing training data mistakenly deemed noisy*. Overall, these results help establish that BoU MR can further improve the multiple models for LO analysis using LSVM.

Once again, using LSVM to improve models is important on EDM datasets that contain relatively large amounts of noise. However, LSVM used only once on a highly diverse datasets may struggle to find concentrations of label noise and could end up—in its pursuit of noise to correct—flagging data near the decision boundary as noisy. Again, BoU MR helps in this regard by allowing for more effective LSVM focusing on the clusters containing large amounts of noise.

5. CONCLUSIONS AND FUTURE WORK

Supervised learning (SL) systems have been used for the analysis of student interactions with learning objects (LOs). SL systems learn a model from previously collected training data that can accurately predict the labels for new data assuming that the training data comes from a single distribution and is relatively clean. Unfortunately, LO data is highly diverse from multiple distributions and likely to contain noise which can limit the effectiveness of single models. Learning multiple models and improving models on troublesome training data is intuitively a good solution. However, identifying areas that capture only data from a single distribution without fitting the noise is far from trivial. We propose a new BoU meta-reasoning (MR) algorithm that starts by breaking up the data into clusters designed to separate troublesome data (incorrect cluster) from data where the model is doing well (correct cluster). Our algorithm then uses a separate model for each cluster. On the incorrect clusters, our algorithm takes steps to improve the model by using feature selection or noise correction on the training data before relearning the model. We have shown empirically on three LO datasets that the BoU multiple models allow for higher predictive accuracy than single models. Furthermore, we have shown that the BoU MR can further improve the models for LO analysis using feature selection and noise correction algorithms. Our results also suggest that BoU MR may also be able improve feature selection and noise correction algorithms—both important for educational data mining. Feature selection used separately on the BoU clusters makes it easier to identify features that are relevant only on certain areas. Further, noise correction focused on the clusters makes it easier to remove noise instead of accidently disrupting the decision boundary.

In this paper, we have established effectiveness of using the BoU multiple models. In the future, we intend to advance our BoU MR down two different lines of research. First, we intend to further investigate how the BoU multiple models can provide independent, high-level guidelines on which type of student interactions are associated with the learning outcomes. We have already taken the first step by showing that multiple models allow for diverse decision trees. The next step is the analysis of the ANN and SVM models using rules extraction, sensitivity analysis, or inverse classification techniques [3]. This research has strong pedagogical implications for students. Such analysis could, in real-time, inform students about the probable success/failure of their study strategies, for example, warning a student during the LO that she may not be spending enough time on the tutorial section. Second, we intend to expand the use of BoU models to other educational data mining (EDM) areas such as intelligent tutoring systems (ITS) and virtual learning platforms (VLP). These areas share some of the same properties that BoU is designed to address (e.g., noise) but have additional properties not found in LOs (e.g., “bags” of instances in VLP). To this end, we intend to evaluate how the BoU models stack up against previous work on learning accurate models in these areas (e.g., Rajibussalim [14] for ITS; Zafra & Ventura [20] for VLP).

6. REFERENCES

[1] Alharbi, A., et al. 2011. An Investigation into the Learning Styles and Self-Regulated Strategies for Computer Science Students. *ASCILITE*, 36-46.

[2] Balatsouka, P., Morris, A., O’Brien, A. 2008. Learning Objects Update: Review and Critical Approach to Content Aggregation. *Educational Technology & Society*, *11* (2), 119-130.

[3] Barbella, D., et al. 2009. Understanding Support Vector Machine Classifications via a Recommender System-Like Approach. *DMIN*, 305-311.

[4] Bernardini, A., Conati, C. 2010. Discovering and Recognizing Student Interaction patterns in Exploratory Learning Environments. *ITS*, 125-134.

[5] Dasgupta, S., Hsu, D. 2008. Hierarchical Sampling for Active Learning. *ICML*, 208-215, 2008.

[6] de Raadt, M., Simon. 2011. My Students Don’t Learn the Way I Do. *ACE*, *114*. 105-112.

[7] Freidman, J., Hastie, T., Tibshirani, R. 2010. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, *33* (1), 1-22.

[8] Kiu, C-C. Lee, C-S. 2007. Learning Object Reusability and Retrieval through Ontological Sharing: A Hybrid Unsupervised Data Mining Approach. *ICALT*, 548-550.

[9] Kulis, B., Basu, S., Dhillon, I., Mooney, R. 2009. Semi-Supervised Graph Clustering: A Kernel Approach. *Machine Learning*, *74* (1), 1-22.

[10] Liu, H., Yu, L. 2005. Towards Integrating Feature Selection Algorithms for Classification and Clustering. *IEEE Trans. On Knowledge and Data Engineering*, *17* (4), 491-502.

[11] Miller, L. et al. 2011. Evaluating the Use of Learning Objects in CS1. *SIGCSE*, 57-62.

[12] Nugent, G., Soh, L-K., Samal, A., Lang, J. 2006. A Placement Test for Computer Science: Design, Implementation, and Analysis. *Computer Science Education*, *16* (1), 19-36.

[13] Pechenizkiy, M., et al. 2006. Class Noise and Supervised Learning in Medical Domains: The effect of feature extraction, *IEEE CBMS*, 708-113.

[14] Rajibussalim. 2010. Mining Students’ Interaction Data from a System that Support Learning by Reflection. *EDM*, 249-256.

[15] Romero, C., Ventura, S., Espejo, P., Hervas, C. 2008. Data Mining Algorithms to Classify Students. *EDM*, 187-191.

[16] Romero, C., Ventura, S. 2010. Educational Data Mining; A Review of the State of the Art. *IEEE Trans. On Systems, Man, and Cybernetics*, *40* (6), 601-618.

[17] Segata, N. 2009. FaLKM-lib v1.0: A Library for Fast Local Kernel Machines. University of Trento, Italy.

[18] Segata, N., Blanzieri, E. Fast and Scalable Local Kernel Machines. 2010. *JMLR*, 1883-1926.

[19] Witten, I., Frank, E., Hall, M. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier.

[20] Zafra, A., Ventura, S. 2009. Predicting Student Grades in Learning Management Systems with Multiple Instance Genetic Programming. *EDM*, 309-318.